

NAME-RAJVEER ARORA, UID-23BCC70023, SUB-ADBMS

EXPERIMENT-02

- **AIM:-** To design a normalized academic schema (up to 3NF) for managing departments and their courses, populate it with meaningful sample data, query departments offering more than two courses using a subquery, and implement access control using Data Control Language (DCL).

- **THEORY:-**

□ **Normalization (3NF):** The Third Normal Form eliminates transitive dependencies. A relation is in 3NF if it is in 2NF and no transitive functional dependency exists between non-prime attributes.

□ **Relational Model Design:**

□ **Departments** table holds unique department data.

□ **Courses** table associates each course with exactly one department via a foreign key.

□ **Subqueries:**

A subquery is a query nested inside another SQL query. It helps in filtering, transforming, or summarizing data based on related conditions. □

Access Control using DCL:

Data Control Language statements like GRANT manage user privileges.

Granting SELECT access ensures a user can view but not modify data.

- **CODE:-**

-- Drop if exists for clean re-execution

DROP TABLE IF EXISTS Courses;

DROP TABLE IF EXISTS Departments;

-- Create Departments table

```
CREATE TABLE Departments (    dept_id INT
PRIMARY KEY,    dept_name VARCHAR(50)
UNIQUE NOT NULL
);
```

-- Create Courses table

```
CREATE TABLE Courses (    course_id INT
PRIMARY KEY,    course_name
VARCHAR(100) NOT NULL,    dept_id INT
NOT NULL,
FOREIGN KEY (dept_id) REFERENCES Departments(dept_id) ON
```

DELETE CASCADE

);

□ INSERTION OF DATA:

-- Insert Departments

INSERT INTO Departments (dept_id, dept_name) VALUES

(1, 'Computer Science'),

(2, 'Electrical'),

(3, 'Mechanical'),

(4, 'Civil'),

(5, 'Electronics');

-- Insert Courses

INSERT INTO Courses (course_id, course_name, dept_id) VALUES

(101, 'DBMS', 1),

(102, 'Operating Systems', 1),

(103, 'Power Systems', 2),

(104, 'Digital Circuits', 2),

(105, 'Thermodynamics', 3),

(106, 'Fluid Mechanics', 3),

(107, 'Structural Engineering', 4),

(108, 'Surveying', 4),

(109, 'Embedded Systems', 5),

(110, 'VLSI Design', 5);

-- Insert Courses if more than 2 courses

SELECT dept_name

FROM Departments

WHERE dept_id IN (

SELECT dept_id

FROM Courses

GROUP BY dept_id

HAVING COUNT(*) > 2

);

-- Grant Access to the user

GRANT SELECT ON TABLE Courses TO viewer_user;

• OUTPUTS:-

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer shows the database structure for 'CC' and 'postgres'. The main window displays a SQL query in the 'Query' tab. The query is as follows:

```

32 DELETE FROM Departments2
33 FROM Departments2
34 WHERE dept_id IN (
35   SELECT dept_id
36   FROM Courses2
37   GROUP BY dept_id
38   HAVING COUNT(*) > 2
39 );
40 CREATE ROLE viewer_user LOGIN PASSWORD '@Aru_316';
41
42 GRANT SELECT ON TABLE Courses2 TO viewer_user;
43
44 SELECT * FROM Courses2;
45

```

Below the query, the 'Data Output' tab shows the results of the 'SELECT * FROM Courses2;' query. The results are displayed in a table with the following columns: course_id [PK] integer, course_name character varying (100), and deptId integer. The table contains 10 rows of data.

course_id [PK] integer	course_name character varying (100)	deptId integer
1	DBMS	1
2	Operating Systems	1
3	Power Systems	2
4	Digital Circuits	2
5	Thermodynamics	3
6	Fluid Mechanics	3
7	Structural Engineering	4
8	Surveying	4
9	Embedded Systems	5
10		

The status bar at the bottom indicates 'Total rows: 10' and 'Query complete 00:00:00.204'.

• LEARNING OUTCOMES:-

- Understand and apply **3NF normalization** in database design. □ Use **foreign key constraints** to maintain referential integrity.
- Write **subqueries** using GROUP BY and HAVING to analyze relationships.
- Implement **access control** using GRANT statements in PostgreSQL.
- Handle **real-world schema modeling** and data organization tasks efficiently.